Minimization of Sewage Network Overflow

Bernat Joseph-Duran*[1], Michael N. Jung†[2], Carlos Ocampo-Martinez‡[1], Sebastian Sager§[3] and Gabriela Cembrano¶[1]

[1]Institut de Robòtica i Informàtica Industrial (CSIC-UPC), Universitat Politècnica de Catalunya.
[2]Interdisciplinary Center for Scientific Computation, University of Heidelberg.
[3]Institute for Mathematical Optimization, Otto-von-Guericke Universität Magdeburg.

October, 2013

Abstract

We are interested in the optimal control of sewage networks. It is of high public interest to minimize the overflow of sewage onto the streets and to the natural environment that may occur during periods of high rain. The assumption of linear flow in a discrete time setting has proven to be adequate for the practical control of larger systems. However, the possibility of overflow introduces a nonlinear and nondifferentiable element to the formulation, by means of a maximum of linear terms. This particular challenge can be addressed by smoothing methods that result in a *nonlinear program* (NLP) or by logical constraints that result in a *mixed integer linear program* (MILP). We discuss both approaches and present a novel tailored branch-and-bound algorithm that outperforms competing methods from the literature for a set of realistic rain scenarios.

1 Introduction

Combined sewer networks are present in many large cities all over the world. These networks carry both wastewater and storm water together. During low to moderate rain events, this water is carried to wastewater treatment plants, where it is treated before being released to the receiving environment (usually a river or the sea). However, during heavy-rain events the network capacity can be easily overloaded, causing urban surface flooding as well as untreated water discharges to the environment, known as *combined sewer overflows* (CSO).

To avoid these unwanted discharges, retention tanks are usually built along the network to store the water during the peak rain intensity periods and later release it at lower flow rates. Since these infrastructures are clearly expensive and difficult to locate in urban areas, its efficient operation has become a topic of major interest.

Due to the network structure as well as the uncertain distribution of the rain inflows, global real-time control through network monitoring and rain intensity forecasts is regarded as the best control option (Schütze et al., 2004). This approach has been studied in Gelormino and Ricker (1994); Marinaki and Papageorgiou (1998, 2001, 2005); Pleau et al. (2005); Cembrano et al. (2004); Puig et al. (2009); Ocampo-Martinez and Puig (2010); Ocampo-Martinez (2011) among others.

In this work, a network model is presented to be used in an optimal control strategy to minimize unwanted sewage discharges. Having in mind that a real-time control approach would require to solve the optimal control problem at every time step (in a model predictive control strategy), the main contribution of this paper focuses on discussing four different formulations and solving procedures for the obtained optimization problems in order to determine the best option in terms of solving time.

The paper is organized as follows. In Section 2, after a short review of the physical model, we present a generic modeling approach for the catchment flow and overflow in sewage networks and

*bjoseph@iri.upc.edu
†michael.jung@iwr.uni-heidelberg.de
‡cocampo@iri.upc.edu
§sager@ovgu.de
¶cembrano@iri.upc.edu

also specify our choices for the flow system, which result in a linear system for the network flows and a piecewise linear equations system for the overflows. In Section 3, we present four ways to handle the piecewise linear equation system: firstly, we use a nonlinear formulation, which smooths the nondifferentiable kinks. Secondly, we reformulate the system using the *mixed logical dynamical* (MLD) formulation, which results in an MILP. Thirdly, we present the *constraint branching algorithm*, which is a tailored branch-and-bound algorithm based on a linear relaxation of the piecewise linear equations system and then branches on these constraints rather than variables. The implementations of the two latter approaches are described in Section 4 and their computational results are compared. Finally, we consider the *general disjunctive programming* (GDP) approach, which is a formal generalization of the MLD approach and which can be resolved in a second way. Finally, we draw the main conclusions and give an outlook in Section 5.

2 Sewer network modeling

The main difference among the aforementioned studies is the mathematical model that is used to describe the network dynamics. Sewer networks are usually non-pressurized ones, i.e., water flows only due to the effect of gravity. According to Marinaki and Papageorgiou (1998) and Schütze et al. (2002), the preferred physical model in sewer network simulation software for open-channel flow is based on the one-dimensional Saint-Venant equations (see Section 2.1) or one of its simplified forms (mainly the diffusive wave equations).

Although some works in the control of irrigation canals field deal with the control of canal reaches described by the Saint-Venant equations (Coron et al. (1999); Leugering and Schmidt (2002); de Halleux et al. (2003); Coron et al. (2007), among others), they provide mostly theoretical results whose application would be limited to simple network topologies. On the other hand, the linearized Saint-Venant equations are the base of many other works within the irrigation canals field, where usually one single straight canal with several hydraulic structures is considered (Litrico and Fromion (2009); Malaterre and Baume (1998) and the references therein). For real-time optimization-based control of sewer networks, the size and topological complexity of the networks have required the development of faster simplified models for optimal control purposes (Rauch et al., 2002).

Simplified models must be able to capture the most notable features of the dynamics at computationally acceptable efforts, compatible with the real-time computation of the control strategies. In an optimal control context, the simplified model equations are numerically solved inside each optimizer iteration (shooting approach), or appear directly as constraints of an optimization problem (collocation approach). In both cases, the problem's features are transferred to the discretized optimization problem (e.g., number of variables, convexity, linearity, presence of integer/Boolean variables). The discretization/integration of the Saint-Venant equations for the complete sewage network would lead to an optimization problem not solvable in a real-time control framework, if even solvable at all. Although the Saint-Venant equations do not appear directly in the optimization problems, some strategies run simulations using the Saint-Venant model each time-step to update the simplified model (Darsono and Labadie, 2007).

To run the models, both for simulation and control, it is assumed that a short-term prediction of the rain intensity is available at all points of the network.

2.1 Physical model

The commonly used physical model for water motion in sewer networks is based on the 1D Saint-Venant equations with constant cross-sectional area and constant bed slope for each channel (Marinaki and Papageorgiou, 2005; Schütze et al., 2002; Ocampo-Martinez, 2011; Rauch et al., 2002). These equations are hyperbolic nonlinear partial differential equations (PDEs) relating the flow and water level in an open channel/sewer:

$$\partial_t A(x,t) + \partial_x Q(x,t) = 0,$$

$$\partial_t Q(x,t) + \partial_x \left(\frac{Q^2(x,t)}{A(x,t)} \right) + g\, A(x,t)\, \partial_x Y(x,t) = g\, A(x,t)\, (S_b(x) - S_f(x,t)),$$

where

- x is the longitudinal coordinate [m],

- t is the time [s],
- $Q(x,t)$ is the flow $[\frac{\mathrm{m}^3}{\mathrm{s}}]$,
- $A(x,t)$ is the cross-sectional area of the flow $[\mathrm{m}^2]$,
- $Y(x,t)$ is the water level [m],
- S_b is the bed slope [dimensionless],
- $S_f(x,t)$ is the friction slope [dimensionless], e.g. approximated by the Manning formula (Chow, 1959; Chaudhry, 2008; Litrico and Fromion, 2009):

$$S_f = \frac{Q^2 n^2}{A^2 R^{4/3}},$$

  with n denoting the Manning coefficient (depending on the channel physical properties) and R the hydraulic radius,
- g the gravitational acceleration $[\frac{\mathrm{m}}{\mathrm{s}^2}]$.

The equations are applied to each sewer and coupled by means of internal and boundary conditions defined in joints, sewer geometry changes and hydraulic structures. Typical hydraulic structures are overflow points, weirs, gates, and reservoirs. Equations for hydraulic structures relate the flows and/or water levels up- and downstream of the structure. These connecting equations imply that the dynamics of the network need to be solved as a single system, and cannot be solved independently for each sewer. Therefore, the computations become very demanding for big networks with complex topologies.

2.2 Simplified network model

The model presented in the following is based on the so-called *virtual tank* model. This model was first developed by Gelormino and Ricker (1994) to be used as a control model for optimization-based control of large-scale sewer networks. It is a *conceptual model* that does not deal with flows in the individual network sewers but it splits the network into catchment areas and approximates the flows between these areas. The volume contained in each catchment area is modeled like the volume in a tank, which gives the model its name. According to Ocampo-Martinez (2011), a virtual tank is defined as follows:

> "At any given time, let the *virtual tank* be a storage element that represents the total volume of sewage inside the sewer mains associated with a determined sub-catchment of a given sewer network. The sewage volume is computed via the mass balance of the stored volume, the inflows and the outflows related to the sewage mains, and the equivalent inflow associated with rainwater."

Recently, the *virtual tank* model has been the basis for several studies to verify its usefulness as a control model (Cembrano et al., 2004; Puig et al., 2009; Ocampo-Martinez and Puig, 2010; Ocampo-Martinez, 2011). They use closed-loop simulations against a commercial simulator based on the Saint-Venant equations to obtain performance results of the controllers based on the *virtual tank* model. The virtual tank model is also the control model behind the development of the sewer network control tool CORAL (Spanish for Optimal Control of Sewer Networks) (Puig et al., 2009). A novel way to deal with CSO as well as surface flooding based on the work of Ocampo-Martinez (2011) is also included in the present study.

From the control point of view, the virtual tank model is designed to be used as an upper layer global controller. The flows computed by the optimal control problem based on the model are used as setpoints for local PID controllers at the network gates that regulate the gate position as a function of the measured outflow.

In the *virtual tank* model approach, sewer networks consist of several elements, which are described in the following. For water storage, there are water retention *tanks* built by the network operator and so-called *virtual tanks*, each of which representing a set of sewage collectors for a specific zone of the city. These tanks are taken together in the set T – their main difference is that for tanks usually the in- and outflow can be controlled, whereas for virtual tanks there are not necessarily controllable flows. There is a treatment plant to clean the sewage, where, optimally, all the sewage should be processed. Then, there are *sewers* to connect the different tanks and virtual tanks, which can be partly controlled with pumps and valves. In some sewers,

there are *redirection gates* to manipulate and redirect the flow. Other sewers are connected by simple junctions. Both these structures are treated as tanks with maximum volume of 0 and where all inflow is directly forwarded as outflow. These sewage networks can be displayed as directed graphs. A conceptual example of such a network is displayed in Figure 1.

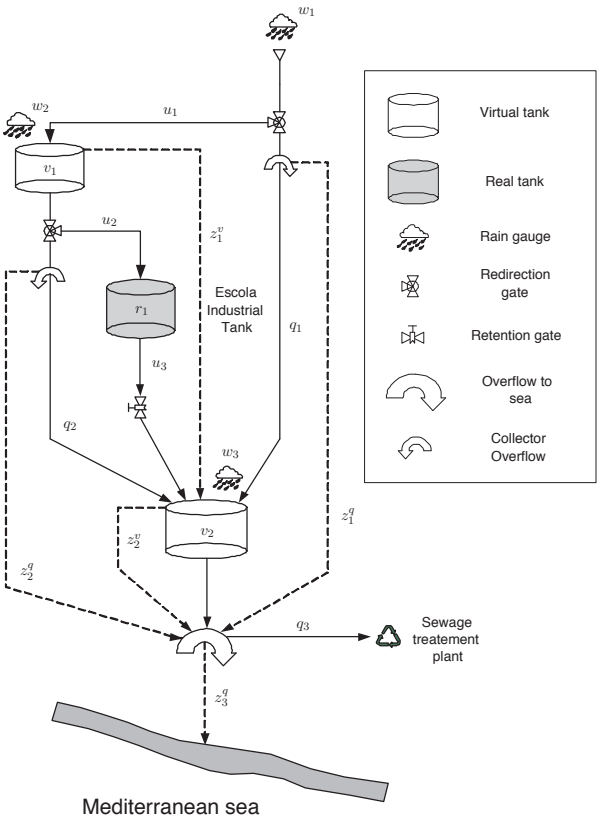


Figure 1: A small example network with two virtual tanks v_1, v_2 and one retention tank r_1. Flow paths that appear due to overflow are represented as dashed connections.

During normal operation, the network can easily transport the sewage towards the treatment plant. However, in the presence of heavy rain, it may happen that there exists no viable flow path for the network to process all the incoming water. In these scenarios, overflow happens – flow paths appear, which were not present before and depend on the system state and inputs. The system states are the volumes v_i in each tank and virtual tank, $i \in T$. We also need to account the flows $q_{i,j}$ through the sewers as dependent variables. However, some of the flows may be directly controlled with valves or pumps, $(i,j) \in C$, while others remain uncontrolled, $(i,j) \in U$.

For notational simplicity, we introduce the accumulated inflow $Q_i^{\text{in}}(t_j)$ for each node $i \in T$ at time step j. This also takes into account the inputs into the system – rain as well as normal sewage – given by $w_i(t_j)$:

$$Q_i^{\text{in}}(t_j) = w_i(t_j) + \sum_{\substack{k \in T: \\ (k,i) \in U \cup C}} q_{k,i}(t_j).$$

For each tank i of the system, there is only one regular outflow Q_i^{out} – controllable or not. For the uncontrolled outflows of the tanks due to gravity, we use the following assumption: according to the virtual tank model (Puig et al., 2009) we assume the uncontrolled outflows to be linear in the amount of water:

$$Q_i^{\text{out}}(t_j) = \beta_i \, v_i(t_j), \tag{1}$$

where the parameter β_i is obtained from historical sensor data, or to be calibrated online in a real-time control approach.

A more precise formulation would consider the water pressure in both nodes adjacent to the sewer as well as the friction along it. However, this formulation employs nonlinearities in the model and its contribution is neglectable in our setting. Thus, it shall not be considered in this work. The outflows that completely exit the system are the desired flow into the treatment plant and undesired overflows into the surrounding area polluting the environment.

The system is governed by the mass conservation law for each node i:

$$\dot{v}_i(t) = Q_i^{\text{in}}(t) - Q_i^{\text{out}}(t),$$

with the cumulative inflow $Q_i^{\text{in}}(t)$ and the outflow $Q_i^{\text{out}}(t)$.

This system is discretized with an explicit Euler method on an equidistant time grid with step length Δt and n_t time intervals to obtain a discrete-time linear flow system

$$v_i(t_{j+1}) = v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right), \quad i \in T, j = 1, \dots, n_t - 1, \tag{2}$$

where $t_j = j\Delta t$ represents the j-th time step. For abbreviation, we introduce the set D of discretization indices

$$D := \{1, \dots, n_t - 1\}.$$

All the controlled flows are limited due to physical constraints such as sewer sizes and pump capacities. We assume that the volumes of the real retention tanks are limited as well and they cannot overflow since they are often placed underground and not connected to the surface. Their inflows are hence always controlled to prevent overflow, which physically could happen. In real applications, an overflow emergency mechanism is always present in case there is a malfunction in the controlled devices, but we should not deal with this special situations in this paper.

2.3 Overflow modeling

As mentioned before, overflow happens if the volume obtained by the flow system exceeds the maximum capacity $v_i^{\max}$ of the node i. Joints of different sewers are treated as tanks with maximum capacity $v_i^{\max} = 0$. In case of the flows being in the physical limits, no overflow happens, but if they exceed the physical limits, all the excess amount is considered as overflow. Overflows in each virtual tank and sewer joint can be redirected to other virtual tanks, i.e., to another sewer catchment, or to the receiving environment (usually a river or the sea). Only in the latter case, the overflow volume leaves the network permanently.

As soon as an overflow $z_i(t)$ appears, the corresponding node's mass conservation equation changes to

$$\dot{v}_i(t) = Q_i^{\text{in}}(t) - Q_i^{\text{out}}(t) - z_i(t). \tag{3}$$

All the overflow from one node i flows to the same target node j. For ease of notation, define the set Z as the set of pairs of tank indices for which an overflow path exists from tank i to tank j. The accumulated inflow is adapted to include inflows due to overflow

$$Q_i^{\text{in}}(t_j) = w_i(t_j) + \sum_{\substack{k \in T: \\ (k,i) \in U \cup C}} q_{k,i}(t_j) + \sum_{\substack{k \in T: \\ (k,i) \in Z}} z_k(t_j).$$

The overflow z_i can be modeled with logical decisions

$$\begin{aligned} \text{IF} \quad & v_i^{\max} \leq v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) \\ \text{THEN} \quad & z_i(t_j) = \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right) \\ \text{ELSE} \quad & z_i(t_j) = 0, \end{aligned} \tag{4}$$

where the THEN expression together with the discretized version of (3), i.e.,

$$v_i(t_{j+1}) = v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j) - z_i(t_j)\right), \ i \in T, j \in D, \tag{5}$$

also sets $v_i(t_{j+1}) = v_i^{\max}$. This approach is further explained in Section 3.2.

Another modeling approach uses a disjunction as later described in Section 3.4

$$\begin{bmatrix} \lambda_i(t_j) = 0 \\ z_i(t_j) = 0 \end{bmatrix} \vee \begin{bmatrix} \lambda_i(t_j) = 1 \\ v_i^{\max} \leq v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j) \right) \\ z_i(t_j) = \dfrac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j) \right) - v_i^{\max} \right) \end{bmatrix}, \quad i \in T, j \in D. \quad (6)$$

Finally, the overflow can be modeled directly with the maximum function

$$z_i(t_j) := \frac{1}{\Delta t} \max \left\{ 0, v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j) \right) - v_i^{\max} \right\}, \quad i \in T, j \in D. \quad (7)$$

Then, the constraints are piecewise affine equations and thus nonlinear and nondifferentiable. Therefore, each system directly containing these constraints becomes not only nonlinear but also nonconvex and nondifferentiable. One such constraint is displayed in Figure 2.

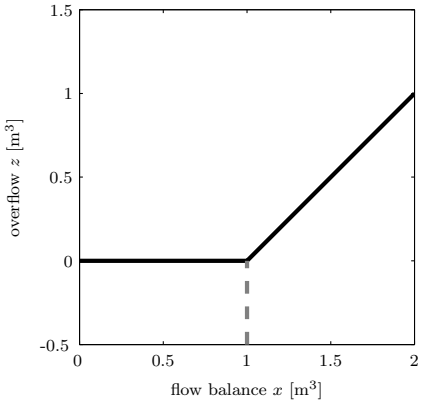


Figure 2: The overflow function $z(x) = \max\{0, x - 1\}$ with maximum capacity $v^{\max} = 1$ and water volume change x. It is nonlinear, nonconvex and nondifferentiable in the kink at $x = 1$.

Overflows in joints (junctions and redirection gates) are modeled in an analogous way by considering a joint as a tank with zero capacity, and hence constant volume of 0 (also $v_i(\cdot) = 0$). In this case, overflows occur when the inflow to the joint exceeds the maximum outflow. Due to notational simplicity, we omit this part of the formulation for the remainder of this paper since there are no additional insights gained. All formulations directly carry over to these slightly changed conditions. Here, we briefly show how to adapt the logical formulation:

$$\begin{aligned} \text{IF} \quad & Q_i^{\text{in}}(t_j) \geq Q_i^{\max} \\ \text{THEN} \quad & z_i(t_j) = Q_i^{\text{in}}(t_j) - Q_i^{\max} \\ \text{ELSE} \quad & z_i(t_j) = 0, \end{aligned}$$

where $Q_i^{\max}$ is the maximum capacity of the sewers leaving joint i.

The objective of the network operators is to minimize all overflows, because they have two negative impacts: inside the city they disturb the people living there and outside the city, they pollute the environment. However, different weights $\omega_i > 0$ can be associated to different overflow points depending on their geographic location. Therefore, the management objective can be formulated as follows:

$$\min \sum_{j \in D, i \in T} \omega_i \, z_i(t_j). \quad (8)$$

3 Reformulations and methods

In an optimal control setting, control actions are obtained as the result of an optimization problem aimed to minimize the value of an objective function that describes the system performance. The model's dynamics either appear as constraints of this optimization problem or need to be integrated within each optimizer iteration. The formulation of the optimization problem together with the used solving algorithm is of capital importance in terms of the time needed to solve the problem and the quality of the obtained solution.

Going a step further from the optimal control approach, a predictive control strategy solves an optimal control problem at each time step taking advantage of new sensor information of the system and new predictions of the perturbations. Thus, it is of great interest to develop fast solving algorithms that guarantee that the solution is obtained within the available time. This would be the goal of the application to the real world.

This section covers four approaches to handle the given overflow formulation and obtain optimization models that can either be solved by standard algorithms or the algorithm needed is directly specified. The first approach is based on replacing the nondifferentiable overflow equations by differentiable approximations. We obtain an NLP, which can be locally solved by a derivative-based algorithm. The other three approaches are based on the fact that the nonlinearities and nondifferentiabilities are induced by the piecewise linear overflow equation. Piecewise linear models can be formulated to produce linear optimization problems with binary variables and they can then be solved with MILP algorithms. The difference between the last three approaches is that the first and third ones make explicit use of the binary variables to obtain a problem, which can be solved by standard MILP solvers, while the second one makes use of the problem properties to come up with a constraint branching strategy that encodes the logic decisions.

3.1 Nonlinear smoothing

Constraints (7) can be reformulated using smoothing for the nondifferentiabilites. We use hyperbolic smoothing to connect the two arms of the maximum function with a portion of a circle of radius r. The circle center is at (m_1, r) with

$$m_1 := (1 - \sqrt{2})r + v_i^{\max}.$$

The resulting overflow formulation is

$$z_i^r(x(t_j)) = \begin{cases} 0, & \text{if } x_i(t_j) \leq m_1, \\ r - \sqrt{r^2 - (x_i(t_j) - m_1)^2}, & \text{if } x_i(t_j) \in \left(m_1, m_1 + \frac{r}{\sqrt{2}}\right), \\ x_i(t_j) - v_i^{\max}, & \text{if } x_i(t_j) \geq m_1 + \frac{r}{\sqrt{2}}, \end{cases}$$

with

$$x_i(t_j) := \frac{1}{\Delta t} v_i(t_j) + Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j).$$

We obtain the continuously differentiable function $z_i^r(\cdot) \in C^1(\mathbb{R})$, which is sufficiently differentiable for the usual nonlinear solving methods, i.e., active set and interior point based on accumulated first derivative approximations for the Hessian. The change from the previous formulation is displayed in Figure 3.

3.2 Mixed Logical Dynamical system

The MLD system formulation can be used to combine the inherent logic of the overflows with the dynamical control problem. Bemporad and Morari (1999) describe how to use the logical formulation (4) to obtain an MILP with binary variables to model the logical decisions. They employ binary variables λ to capture the logics of the system and provide a set of inequalities to combine the new binary variables with dynamic structure of the continuous states. Under some assumptions, MLD systems have been shown to be equivalent to other system modeling formats including *linear complementarity* systems, *extended linear complementarity systems*, *piecewise affine systems*, and *max-min-plus-scaling* systems (Heemels et al., 2001).

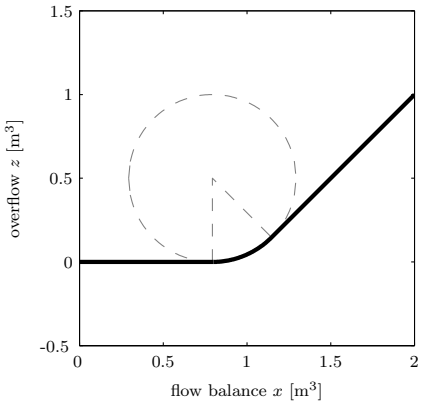


Figure 3: The smoothed function with maximum capacity $v_i^{\max} = 1$, $\Delta t = 1$ and radius $r = 0.5$. Due to smoothing, the nondifferentiable kink is replaced by a differentiable approximation.

As an example, let us consider the following equivalence condition of a binary variable λ and a linear function f with the connection of the two being

$$[\lambda = 1] \leftrightarrow [f(x) \geq 0].$$

This condition is equivalent to the following linear inqualities:

$$\begin{aligned} f(x) &\geq m\,(1-\lambda)\,, \\ f(x) &\leq (M+\epsilon)\,\lambda - \epsilon, \end{aligned}$$

where

$$\begin{aligned} m &:= \min_{x \in \mathcal{D}} f(x), \\ M &:= \max_{x \in \mathcal{D}} f(x), \end{aligned}$$

and ϵ is a small tolerance parameter beyond which the constraint is considered to be violated. If we assume f to be a linear function and variables x to belong to a bounded domain $\mathcal{D}$, m and M can be computed or at least under- and overestimated, respectively, which is enough for the equivalence between the logic statement and the set of inequalities to hold. As usual with Big-M-formulations, the reformulation computationally works better, the smaller the entries of M and the larger the entries of m, as long as they remain valid bounds (Williams, 1999).

In the case of overflow modeling, we use the following logical statements. We define binary variables $\lambda_i(t_j) \in \{0, 1\}$, which model whether there is overflow of node i at time step j. This results in the following conditional system:

$$[\lambda_i(t_j) = 1] \leftrightarrow \left[v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max} \geq 0\right] \tag{9}$$

and

$$\begin{aligned} \text{IF} \quad & \lambda_i(t_j) = 1 \\ \text{THEN} \quad & z_i(t_j) = v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max} \\ \text{ELSE} \quad & z_i(t_j) = 0 \end{aligned} \tag{10}$$

together with the flow equation (5).

IFF-THEN-formulations (9) are put into inequalities with a tolerance parameter ϵ and bounds m_i and M_i, which can be computed using the bounds of the involved variables

$$\begin{aligned} v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max} &\geq m_i\,(1 - \delta_i(t_j)) \\ v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max} &\leq (M_i + \epsilon)\,\delta_i(t_j) - \epsilon. \end{aligned}$$

The IF-THEN-ELSE-formulations (10) are reformulated with

$$z_i(t_j) = \lambda_i(t_j) \left(v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right),$$

which is, for $\lambda_i(t_j) \in \{0,1\}$, equivalent to the linear system

$$\begin{aligned}
&z_i(t_j) \le M_i \lambda_i(t_j),\\
&z_i(t_j) \ge m_i \lambda_i(t_j),\\
&z_i(t_j) \le v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max} - m_i \left(1 - \lambda_i(t_j)\right),\\
&z_i(t_j) \ge v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max} - M_i \left(1 - \lambda_i(t_j)\right),
\end{aligned}$$

with the same m_i and M_i used above.

The reformulation can be automatically done in MATLAB™ with the *Hybrid System DEscription Language* (HYSDEL, see Torrisi and Bemporad (2004)), which is now part of the *Multi-Parametric Toolbox* (MPT, see Kvasnica et al. (2004)). However, HYSDEL does not deal with systems including disturbances, that in our case appear as rain inflow to the network. To solve this problem, rain inflow variables are initially regarded as controlled variables and later the resulting matrices of the MLD format produced by HYSDEL are split to separate the controlled flows from the rain inflow disturbances. As a drawback to this workaround artificial bounds $w_i^{\max}$ on the rain flow variables have to be added.

If we used the nonlinear pressure formulation for the outflow instead of the linearized version, this approach would produce a *mixed-integer nonlinear program* (MINLP). These problems are usually very difficult to solve – especially considering the problem size, which arises from the time discretization together with the network size.

3.3 Constraint branching algorithm

We use the specific problem structure induced by the formulation of overflow to create a branch-and-bound algorithm. It branches on decisions without adding the corresponding variables explicitly to the problem. In contrast to the previously described MLD method, we do not have to add artificial variable bounds and tolerance parameters, which have an impact on the behavior of the algorithm since they slightly disturb the model.

The maximum equation formulation from Section 2.3

$$z_i(t_j) := \frac{1}{\Delta t} \max\{0, v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\}$$

can be relaxed to the linear inequalities

$$z_i(t_j) \ge 0, \tag{11}$$

$$z_i(t_j) \ge \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right). \tag{12}$$

Thereby, we obtain a *linear programming* (LP) problem as the relaxed problem. The meaning of the relaxation is that the controller can choose, during each time step for each node, how much overflow there is – even if in this situation no overflow would actually occur in reality. It can have a positive impact to artificially generate an overflow because it may prevent overflow at another part of the system, which may have worse weights ω_i or may propagate further overflow. However, since overflows $z_i(t_j)$ are to be minimized, for optimal solutions often one of the inequalities (11) and (12) holds with equality.

If the solution of the relaxed problem fulfills for all pairs of inequalities (11) and (12) one of both with equality, then it is already a solution of the original problem. If this is not the case for all overflows, the algorithm has to enforce this behavior. Branching is done on the decision which one of the two inequalities shall be fulfilled with equality for all children in the branching tree. Instead of dichotomy branching on variable values and adding the constraints $x \le \lfloor \bar{x} \rfloor$ or $x \ge \lceil \bar{x} \rceil$ for a fractional variable $\bar{x}$, we add for one branch the constraint

$$z_i(t_j) = 0, \tag{13}$$

and for the second branch the constraint

$$z_i(t_j) = \frac{1}{\Delta t} \left(v_i(t_j) + \Delta t \left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right), \tag{14}$$

if the relaxed solution $\bar{z}$ fulfilled both with ">". This is done until for each overflow, either (13) or (14) holds, which is then a solution of the original problem. This property is enforced through the branching procedure.

We use the following problem formulation OF(A,B) for subproblems at the nodes. We use the sets A and B of index pairs to describe where the overflow has already been fixed for this node:

$$\min \sum_{j\in D, i\in T} \omega_i \, z_i(t_j) \qquad \text{(OF(A,B))}$$

s.t. flow equations and network limits,

$$\begin{aligned}
z_i(t_j) &\geq \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right), && j \in D, i \in T,\\
z_i(t_j) &\geq 0, && j \in D, i \in T,\\
z_i(t_j) &= \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right), && (i,j) \in A,\\
z_i(t_j) &= 0, && (i,j) \in B.
\end{aligned}$$

Each node of the branching tree contains its own version of those two sets to describe the fixed constraints and the general algorithm is outlined in Algorithm 1. This is a simple branching framework, which can easily be upgraded to contain a bounding procedure to obtain a branch-and-bound algorithm. We also experimented with the strategy to get the constraint on which to branch next. The best results were obtained with the intuitive strategy of choosing the earliest overflows.

Algorithm 1: Constraint branching algorithm

Data: Network structure, time discretization, rain inputs.
Result: Optimal solution, i.e. best solution in the solution pool.

Create queue of active tree nodes N and add the empty root node ({}, {}) to N.
while N is not empty **do**
  Choose node n with corresonding fixed constraints (A_n, B_n) by search strategy and remove n from N.
  Solve problem OF(A_n, B_n) and obtain solution variables (z, v, q).
  if solution is feasible **then**
    if $\forall (i,j) \notin A_n \cup B_n$:
    $z_i(t_j) = 0$ *or*
    $z_i(t_j) = \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(Q_i^{in}(t_j) - Q_i^{out}(t_j)\right) - v_i^{max}\right)$
    then
      Add n's solution (z, v, q) to the pool of solutions.
    else
      Find a pair (i,j) such that
      $z_i(t_j) > 0$ and
      $z_i(t_j) > \frac{1}{\Delta t}\left(v_i(t_j) + \Delta t\left(Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)\right) - v_i^{\max}\right)$
      Add two new child nodes to N, which have the following fixed constraints:
      - $(A_n \cup \{(i,j)\}, B_n)$: There is overflow at node i at time step j.
      - $(A_n, B_n \cup \{(i,j)\})$: There is no overflow at node i at time step j.

The presented approach is similar to standard branching, with the difference that we branch on the constraints instead of on the variables. It can be applied to other problems as well. One has to have piecewise continuously differentiable equations and that the relaxation of the equations to inequalities provides a convex (preferably linear) problem. It helps if the violation of the equations is punished in the relaxation – even more so if this is an innate property of the problem. This approach is very dependent on the exact problem structure and to our knowledge has not yet been explored.

Notice that there exists a different kind of constraint branching, as e.g., Ryan-Foster branching for SOS1-constraints (Forrest and Tomlin, 2007), which partitions the feasible set due to special constraint structures.

3.4 General Disjunctive Programming

Instead of the two preceding approaches, overflows can also be reformulated with the *general disjunctive programming* (GDP) framework from Grossmann and Ruiz (2012). The authors use general disjunctions in a first step to model reality and then propose different ways to handle these disjunctions. Using this type of formulation and with M_i and the auxiliary formulation

$$x_i(t_j) = \frac{1}{\Delta t} v_i(t_j) + Q_i^{\text{in}}(t_j) - Q_i^{\text{out}}(t_j)$$

as above, and with upper bounds M_i on the new states x_i, we obtain the problem

$$\begin{aligned}
&\min \sum_{j \in D, i \in T} \omega_i(t_j) z_i(t_j) && (15)\\
&\text{s.t. flow equations and network limits,}\\
&\begin{bmatrix} Y_{i,j} \\ z_i(t_j) = x_i(t_j) - \frac{1}{\Delta t} v_i^{\max} \\ x_i(t_j) \geq \frac{1}{\Delta t} v_i^{\max} \end{bmatrix} \vee \begin{bmatrix} \neg Y_{i,j} \\ z_i(t_j) = 0 \\ x_i(t_j) \leq \frac{1}{\Delta t} v_i^{\max} \end{bmatrix}, \qquad j \in D, i \in T,\\
&Y_{i,j} \in \{\text{true, false}\},\\
&0 \leq x_i(t_j) \leq M_i.
\end{aligned}$$

To obtain an MILP in standard form from this formulation, we can either use a Big-M reformulation, e.g. the MLD formulation of Section 3.2, or a *convex hull* reformulation. The Big-M formulation usually produces weak bounds, whereas the relaxation of the convex hull formulation provides tighter bounds. However, the enhancement effect of the tighter bounds can diminish due to the higher effort needed because of the state duplication. Since the MLD approach of Section 3.2 already provides a Big-M formulation, only the convex hull approach is examined in the following.

For the convex hull reformulation, one needs to duplicate the x-variables for each branch of the disjunction and to introduce the binary variables λ as convex multipliers to combine those different branches. The λ-variables model the decision which branch is taken: the one with overflow ($\lambda = 1$) or the one without ($\lambda = 0$). Using this reformulation we obtain the following MILP:

$$\begin{aligned}
&\min \sum_{j \in D, i \in T} \omega_i(t_j) z_i(t_j) && \text{(GDP)}\\
&\text{s.t.} \quad \text{flow equations and network limits,}\\
&x_i(t_j) = x_i^1(t_j) + x_i^2(t_j),\\
&z_i(t_j) = x_i^1(t_j) - \frac{1}{\Delta t} v_i^{\max} \lambda_i(t_j),\\
&x_i^1(t_j) \in \lambda_i(t_j) \left[\frac{1}{\Delta t} v_i^{\max}, M_i \right], \qquad j \in D, i \in T,\\
&x_i^2(t_j) \in (1 - \lambda_i(t_j)) \left[0, \frac{1}{\Delta t} v_i^{\max} \right],\\
&\lambda_i(t_j) \in \{0, 1\}.
\end{aligned}$$

The difference between the convex hull reformulation, which is a general constraint branching framework, and the *constraint branching Algorithm* 1 presented in Section 3.3 lies in the different relaxations. The relaxation of the hull reformulation by letting $\lambda_i(t_j) \in [0, 1]$ is tighter than the LP relaxation proposed in Section 3.3, which can be seen in Figure 4. However, firstly, as in the MLD formulation, tight bounds M_i are needed to provide a good formulation. For the overflow

problem, these bounds are dependent on the rain scenario and hence need to be recalculated for each instance individually to be accurate. Secondly, the relaxation is only tighter in the upper part of the feasible region, which leads to higher overflow values and is not desired since z is to be minimized.

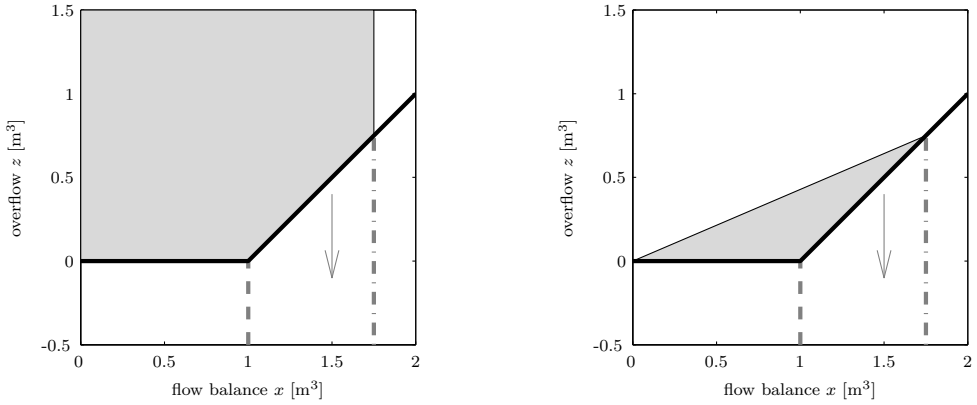


Figure 4: On the left hand side, the LP relaxation from Section 3.3 is drawn, whereas on the right hand side, the projection of the hull formulation's relaxation from (GDP) is drawn. The left dashed line represents the scaled maximum capacity $\frac{1}{\Delta t} v_i^{\max}$ and the right dashed line represents the bound M on the maximum inflow. The arrow represents the objective.

4 Computational results

We used different versions of the approaches presented in the previous sections to compare their performance for the minimization of the overflow of a particular sewage network.

4.1 Test network description

We used a small scale sewage network that is a part of the Barcelona sewer network and for this network data has been provided to build the virtual tank model by the company responsible for the management of the network, CLABSA (Clavegueram de Barcelona S.A.). This network consists of one real tank, 11 virtual tanks and 4 redirection gates, as shown in Figure 5.

For the rain data we used 22 different real rain scenarios to compare the algorithms. This data has also been provided by CLABSA and consists of a selection of rain events occurred in the period between 1996 and 1999, ranging from 4 to 12 hours each. These rain events are representative of the different distribution of the rain intensity both in space and time. Since heavy rain events occur only sporadically, to obtain a larger set of test scenarios, some have been artificially increased by a factor of 2.

The flow $w_i \left[\frac{\mathrm{m}^3}{\mathrm{s}}\right]$ entering the network at virtual tank i is computed from pluviometer data using a conceptual rainfall-runoff model, which is also based on the virtual tank concept, together with a sewage forecast for the catchment area. The rain inflow is obtained by multiplying the rain intensity $I \left[\frac{\mathrm{mm}}{\mathrm{s}}\right]$ with the catchment area $A_i \left[\mathrm{m}^2\right]$ and scaling with a dimensionless ground absorption coefficient ϕ_i (calibrated online) to account for infiltration losses (Puig et al., 2009)

$$w_i(t) = \phi_i \, A_i \, I_i(t).$$

A discussion about the impact of rain forecast on the control performance of the virtual tank model can be found in Cembrano et al. (2002). There, four types of rain predictions for the optimal control problem are tested and compared: perfect prediction, constant prediction (equal to the last measured one), worst-case prediction (increasing or constant intensity, depending on the tendency) and best-case prediction (constant for five minutes and zero afterwards). Taking into account that the predictions are updated at every iteration of the model predictive control

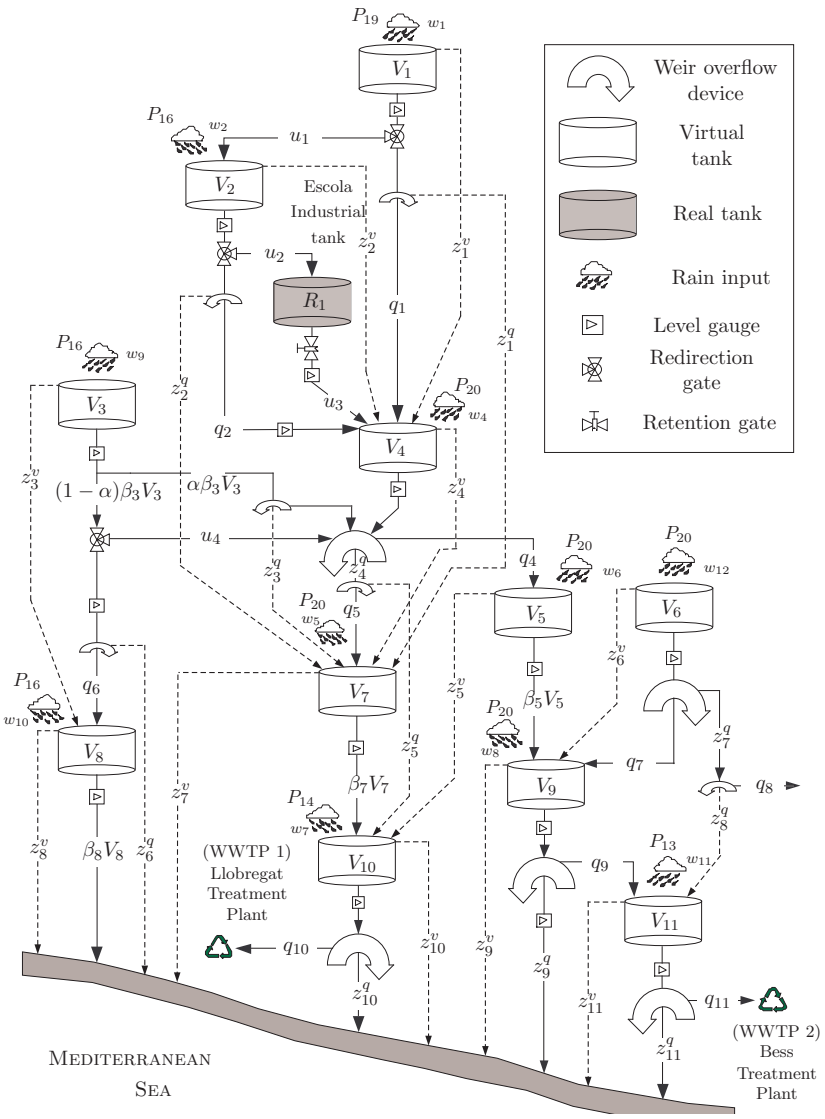


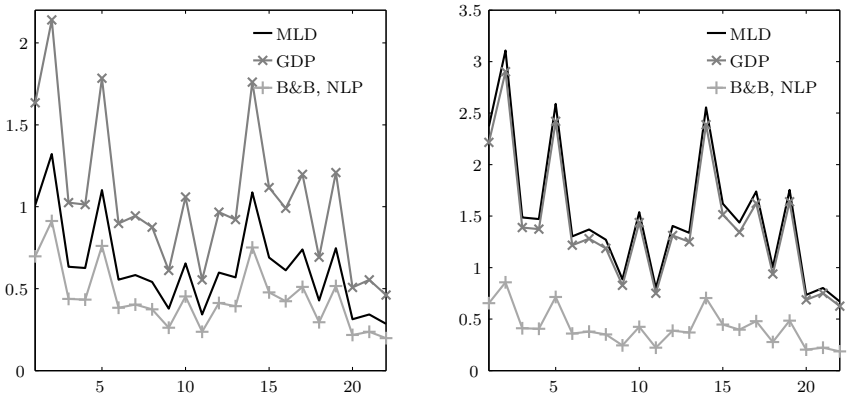
Figure 5: Partial Barcelona sewer network with 12 tanks.

strategy, the reported results show that the constant-rain prediction provides only slightly poorer results than the perfect prediction.

The resulting problem dimensions for the different formulations are displayed in Figure 6. The difference between the instances is the number of time steps of the discretization, i.e., $n_t \in [40, 186]$. The MLD model is always bigger than the model used by the *constraint branching algorithm* as it needs to add integer variables and additional constraints to model the logical decisions. The GDP model requires even more variables than the MLD model, but a little less constraints to formulate the behavior. The number of constraints needed for the constraint branching algorithm is the number of constraints in the root node relaxation. The constraints, which are added in the process of branching, transform inequalities into equations corresponding to their activation. The smoothed, nonlinear formulation has the same number of inequalities as the constraint branching formulation since the constraint branching formulation realizes one part of the maximum term with simple bounds.

4.2 Implementation of the optimization problems

The hyperbolic smoothing approach from Section 3.1 with a nonlinear local algorithm gives results which are worse than the results from the other approaches. Since the formulation is nonconvex, the results are not necessarily optimal in two senses: firstly, the reformulation at



(a) Number of variables (10^4) (b) Number of constraints (10^5)

Figure 6: Problem sizes for the different algorithms on 22 rain scenarios for the sewage network displayed in figure 5.

the kink of the overflow function is not exact and secondly, one would need to use a nonlinear global algorithm to guarantee that the found optimal solution is the true optimal solution, since it is not a convex reformulation. Therefore, purely local algorithms are generally not sufficient to correctly solve the problem.

Nonetheless, we tried to obtain some results for local algorithms since they usually are the subprocedures called in global algorithms. We tested the interior point method IPOPT v3.11 with the HSL linear solver `ma57`. The starting point was the totally uncontrolled network, i.e., all controls and states were set to 0. The solutions differ slightly from the optimal solution obtained by the other approaches due to the smoothing, as shown in Figure 9. However, the solutions were quite close to the global solutions. The deviation is due to the smoothing with $r \sim 10^{-6} \max_{i \in T} \{v_i^{\max}\}$.

For the MLD approach of Section 3.2, we used the HYSDEL package v1.2.8 (Torrisi and Bemporad, 2004) in MATLAB™ v7.7.0 to describe the problem and obtain an MILP. Then, the resulting MILPs are solved with CPLEX™ v12.1.0. The tolerance parameter ϵ, used to transform strict inequalities into non-strict ones, has been set to $\epsilon = 10^{-3}$, which is enough taking into account the variables scale. Additionally, we had to provide artificial upper bounds $w_i^{\max}$ for the rain inputs to allow the transformation of the problem, since they need to be modeled as controlled variables in the MLD framework and then later fixed to the correct perturbation values.

For the *constraint branching algorithm*, we used two different implementations. The first one – a proof of concept – is a very basic implementation of a branch-and-bound algorithm based on the C++ *Standard Template Library* (STL) priority queue for the active nodes of the branching tree with *best-first-search* (BFS). The resulting LPs are solved with CPLEX. However, the algorithm does neither use parallelization for the treatment of nodes nor does it use clever warm-starting as the solution remains feasible in the dual sense. The second implementation uses CPLEX's branch-and-bound framework via callback-routines to use those features. This framework also provides a more sophisticated search strategy for the branching tree, which leads to more LP iterations but also provides a good solution faster.

We reduced the GDP model (GDP) by eliminating the x^2-variables from the model. The needed bounds for the additional x^1-variables were set very coarsely and securely by setting $M_i = v_i^{\max} + W$, where W is the total rain input. The resulting MILPs were solved with CPLEX.

All the problems were solved with CPLEX's standard settings.

4.3 Computational results

The computational results were obtained on a machine with an Intel dual core CPU with 2.66GHz and 8GB RAM. They are displayed in Figures 7–9.

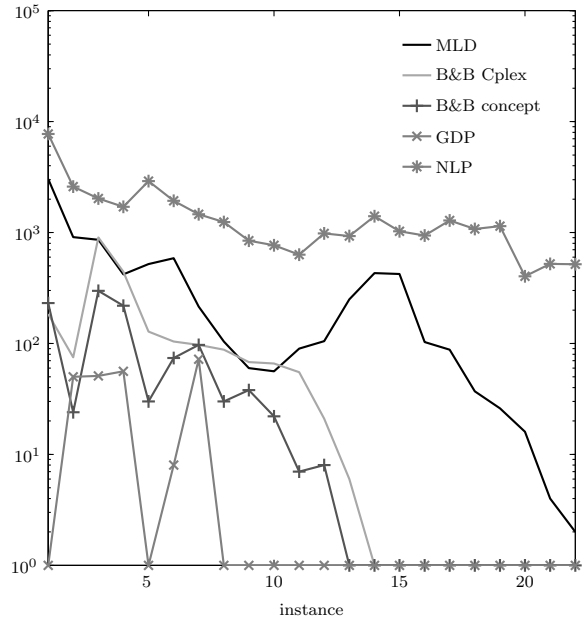


Figure 7: Iteration numbers (log) for the different algorithms.

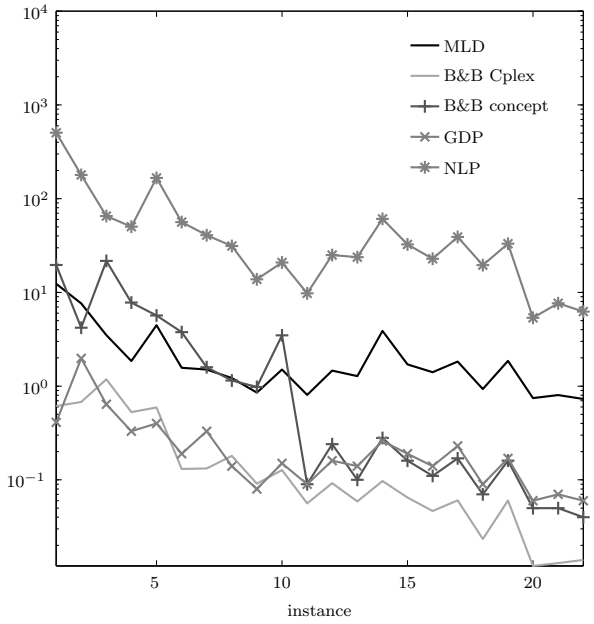


Figure 8: Computational times (log) for the different algorithms.

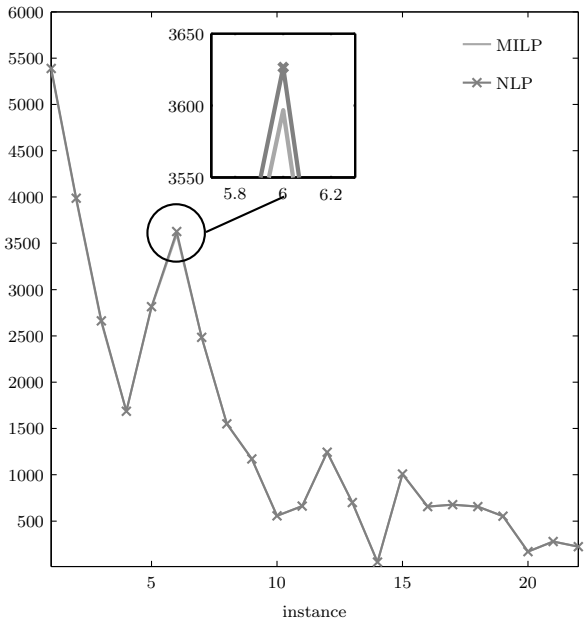


Figure 9: Objective values, i.e., total overflow [m^3], for the linear formulations and the nonlinear formulation.

In the figures, it can generally be seen that the problem difficulty does not necessarily coincide to the problem size. The difficulty is more dependent on the amount of overflow happening in the optimal solution in combination with the problem size.

IPOPT solves all instances but one, although the nonlinear model is nonconvex. Only instance 1 could not be solved to an acceptable level and got stuck in a point of local infeasibility. The objective values are always a little worse (but less than 1% worse) than the objective values of the other algorithms even though the solution is almost identical. This is originated in the smoothing technique used.

Comparing the iteration numbers, c.f. Figure 7, the GDP approach is clearly superior to the other approaches solving most problems in the root node, i.e., 17 of the 22 scenarios. Both constraint branching algorithms compare favorably with the MLD approach with regard to the iteration numbers. They also solve about half the problems in the root node, whereas the MLD approach does not solve any problem in the root. This indicates that the GDP algorithm and the constraint branching algorithm provide much better scalability to larger networks than the MLD approach. Notice that the MLD and GDP approaches use binary variables in the models and can hence use cutting planes, e.g. Mixed-Integer Rounding (MIR)-cuts, to tighten the feasible region, whereas the constraint branching formulation only indirectly introduces those binary variables. For these models, the iteration numbers are the numbers of nodes solved in the branch-and-bound framework. The nonlinear model's iterations are Newton-type iterations of the interior point method IPOPT. The comparison of iterations for the nonlinear solver in comparison to branch-and-bound-based algorithms does not make much sense. However, we can observe that the effort needed is more stable for the nonlinear approach than for the branch-and-bound-based algorithms.

With regard to the computational times, the iteration results do not directly carry over, but additionally the problems' dimensions as well as the implementations have to be taken into account. First, we compare both constraint branching schemes. The more sophisticated approach is about one order of magnitude faster than the simple scheme, which is due to hot-starting and parallelization on two cores. Yet, it needs more iterations, which also comes from the parallelization and from the different search strategy, which incorporates diving to provide a fast near-optimal solution. The simple realization is quite slow and sometimes needs even more time than the MLD realization, which has to solve larger problems. The more sophisticated implementation overcomes this and is almost always the fastest procedure. The GDP approach is slower than the constraint branching algorithm since each iteration (especially the root node) is more costly. One part of this higher effort is due to the larger problem dimensions, but the

time spent in heuristics and cutting plane procedures to tighten the bounds also increase the effort per node. The constraint branching algorithm cannot apply these procedures due to no incorporated binary variables. For IPOPT, the overall computational times are 2–3 orders of magnitude worse than the computational times of the other approaches. Even for these small test instances, the algorithm comes much closer to the overall time limit of 5 minutes – and for instance 1, it takes more than 5 minutes to arrive at the local minimizer of feasibility.

Overall, we can state that the GDP approach takes a little longer than the constraint branching algorithm but the bounds produced by the procedure are tighter. Additionally, the obtained MILP formulation allows the usage of standard algorithms, which can tighten the formulation even further with cutting planes. Therefore, 17 of the 22 test instances were already solved in the root node, in comparison to 12 for Algorithm 1, and the number of iterations were in all but 1 instance the smallest for the GDP approach.

5 Conclusions

We presented a piecewise linear model for overflows in sewage water networks. This model is used in an optimal control approach to minimize overflows in a test network. The optimal control problem is reformulated in four ways to produce equivalent optimization problems, which have been compared computationally. Resulting from these different formulations, we provided several implementations that solve the given instances in the desired five minute time window. The considered problems optimize the controls for the network with rain and water forecasts of almost two days taken into account.

The hyperbolic smoothing approach from Section 3.1 did not provide the desired results. It converged in all but one instance to the global optimum but it needed much more time to find the optimum than the other algorithms.

The GDP approach has some advantages in comparison to the other approaches: It is an easily applied standard modeling approach with no additional thought process involved. It provides much tighter bounds than the constraint branching algorithm and the MLD approach. This probably makes it scale better toward bigger problems even though the computational times are worse than those of the constraint branching algorithm.

Overall, the GDP's perspective formulation seems to be very well suited for disjunctions of small size – in this case, each disjunction has only two disjuncts. It preserves the linearity of the constraints in the disjunctions, which makes the approach especially suited for linear models. The MLD's Big-M formulation compares unfavorably. Although, it preserves linearity and is as easily implemented, the bounds provided by the relaxation are much worse and hence the resulting algorithm performs much worse. The tailored branch-and-bound algorithms provide an alternative but need to exploit the special problem structure to be able to compete with the generally applicable GDP approach with perspective functions. It is no new discovery but still a useful reminder though that a problem specific tailored branch-and-bound algorithm can provide very good results and can outperform the standard approaches, which is especially needed in real-time applications. Additionally, its implementation usually provides deeper insight into the problem and its structures.

There remains the open question of the application into a moving horizon framework. For the test instances, the complete problem could be solved within the time limit but for larger instances, this might not be the case anymore. When the horizon shifts, the contorl of the past problem's first time step is applied and shifted out of the horizon. The old time steps $1, \dots, m-1$ are shifted to become the new problem's time steps $0, \dots, m-2$, and a new time step $m-1$ is created with new forecast data. It is certainly possible to start in the old solution for the remaining time steps and to enumerate all possible branches for the newly created time step. This gives a good initial solution if the new forecast data does not deviate too much from the old forecast data and if the model correctly represents the real system and not only an approximation. It remains open whether it is possible to re-use some of the branching information from the previous runs. The application of *model predictive control techniques* to optimal control problems in Diehl (2002); Kirches (2011) can be taken as a vantage point for future research.

Even though the computational times of IPOPT are much worse than those of the other algorithms, the starting point plays an important role. While we chose an all-0 starting point, in a moving horizon framework, the algorithm could start much closer to the optimum. This would probably speed-up its convergence by a large amount. This question should be studied

when the whole process is transferred to a MPC framework, where the solutions of the past time steps are used to initialize the new starting point.

All of the presented approaches generally remain applicable, if the linear gravity driven flow is replaced with a pressure dependent equation, which also considers friction, to augment the model. However, the MLD and GDP approaches would produce MINLPs. These are often very hard to solve and the state-of-the-art software is not comparably reliable and fast as, e.g., CPLEX in the MILP case. On the other hand, the constraint branching approach would give a branching framework with nonlinear subproblems that is already a tailored solution to an MINLP and that seems at first glance at least as solvable.

Acknowledgements

This work has been partially funded by the EU Project EFFINET (FP7-ICT-2011-8-31855) and DGR of Generalitat de Catalunya (SAC group Ref. 2009/SGR/1491). Financial support of the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences and of the EU project EMBOCON under grant FP7-ICT-2009-4 248940 is gratefully acknowledged.

The cooperation of CLABSA (Clavegueram de Barcelona S.A.) in providing data and useful guidance is also gratefully acknowledged.

References

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427.

Cembrano, G., Figueras, J., Quevedo, J., Puig, V., Salamero, M., and Martí, J. (2002). Global control of barcelona sewerage system for environment protection. In *15th IFAC World Congress*, pages 1–6. Elsevier Science.

Cembrano, G., Quevedo, J., Salamero, M., Puig, V., Figueras, J., and Martí, J. (2004). Optimal control of urban drainage systems: a case study. *Control Engineering Practice*, 12(1):1–9.

Chaudhry, M. H. (2008). *Open-Channel Flow*. Springer, New York.

Chow, V. T. (1959). *Open-Channel Hydraulics*. McGraw-Hill, New York.

Coron, J.-M., d'Andrea Novel, B., and Bastin, G. (1999). A lyapunov approach to control irrigation canals modeled by saint-venant equations. In *European Control Conference*.

Coron, J.-M., d'Andrea Novel, B., and Bastin, G. (2007). A strict lyapunov function for boundary control of hyperbolic systems of conservation laws. *Automatic Control, IEEE Transactions on*, 52(1):2–11.

CPLEX™ (2009). *version 12.1 (2009)*. IBM ILOG, Sunnyvale, California.

Darsono, S. and Labadie, J. (2007). Neural-optimal control algorithm for real-time regulation of in-line storage in combined sewer systems. *Environmental Modelling & Software*, 22:1349–1361.

de Halleux, J., Prieur, C., Coron, J.-M., d'Andréa Novel, B., and Bastin, G. (2003). Boundary feedback control in networks of open channels. *Automatica*, 39(8):1365–1376.

Diehl, M. (2002). *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag.

Forrest, J. and Tomlin, J. (2007). Branch and bound, integer, and non-integer programming. *Annals of Operations Research*, 149(1):81–87.

Gelormino, M. and Ricker, N. (1994). Model-predictive control of a combined sewer system. *International Journal of Control*, 59(3):793–816.

Grossmann, I. E. and Ruiz, J. P. (2012). Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization. In Lee, J. and Leyffer, S., editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, chapter 4, pages 93–116.

Heemels, W., Schutter, B. D., and Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091.

Kirches, C. (2011). *Fast Numerical Methods for Mixed-Integer Nonlinear Model-Predictive Control*. Advances in Numerical Mathematics. Springer Vieweg.

Kvasnica, M., Grieder, P., and Baotić, M. (2004). Multi-Parametric Toolbox (MPT).

Leugering, G. and Schmidt, J. (2002). On the modelling and stabilization of flows in networks of open canals. *SIAM journal on control and optimization*, 41(1):164–180.

Litrico, X. and Fromion, V. (2009). *Modelling and Control of Hydrosystems*. Springer, London.

Malaterre, P.-O. and Baume, J.-P. (1998). Modeling and regulation of irrigation canals: existing applications and ongoing researches. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 4, pages 3850–3855. IEEE.

Marinaki, M. and Papageorgiou, M. (1998). Nonlinear optimal flow control for sewer networks. *Proceedings of the American Control Conference*, 2:1289–1293. Philadelphia, Pennsylvania.

Marinaki, M. and Papageorgiou, M. (2001). Rolling-horizon optimal control of sewer networks. *Proceedings of the IEEE International Conference on Control Applications*, pages 594–599. México City, México.

Marinaki, M. and Papageorgiou, M. (2005). *Optimal Real-time Control of Sewer Networks*. Springer, London.

MATLAB™ (2008). *version 7.7.0 (R2008b)*. The MathWorks Inc., Natick, Massachusetts.

Ocampo-Martinez, C. (2011). *Model predictive control of wastewater systems*. Advances in industrial control. Springer, London.

Ocampo-Martinez, C. and Puig, V. (2010). Piece-wise linear functions-based model predictive control of large-scale sewage systems. *IET Control Theory and Applications*, 4(9):1581–1593.

Pleau, M., Colas, H., Lavallée, P., Pelletier, G., and Bonin, R. (2005). Global optimal real-time control of the Quebec urban drainage system. *Environmental Modelling and Software*, 20(4):401–413.

Puig, V., Cembrano, G., Romera, J., Quevedo, J., Aznar, B., Ramón, G., and Cabot, J. (2009). Predictive optimal control of sewer networks using CORAL tool: application to Riera Blanca catchment in Barcelona. *Water Science and Technology*, 60(4):869–878.

Rauch, W., Bertrand-Krajewski, J.-L., Krebs, P., Mark, O., Schilling, W., Schütze, M., and Vanrolleghem, P. (2002). Mathematical modelling of integrated urban drainage systems. *Water science and technology*, 45(3):81–94.

Schütze, M., Butler, D., and M.B., B. (2002). *Modelling, Simulation and Control of Urban Wastewater Systems*. Springer, London.

Schütze, M., Campisano, A., Colas, H., Schilling, W., and Vanrolleghem, P. (2004). Real time control of urban wastewater systems - where do we stand today? *Journal of Hydrology*, 299(3–4):335–348.

Torrisi, D. and Bemporad, A. (2004). HYSDEL – A tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12(2):235–249.

Williams, H. (1999). *Model building in mathematical programming*. Wiley, New York.